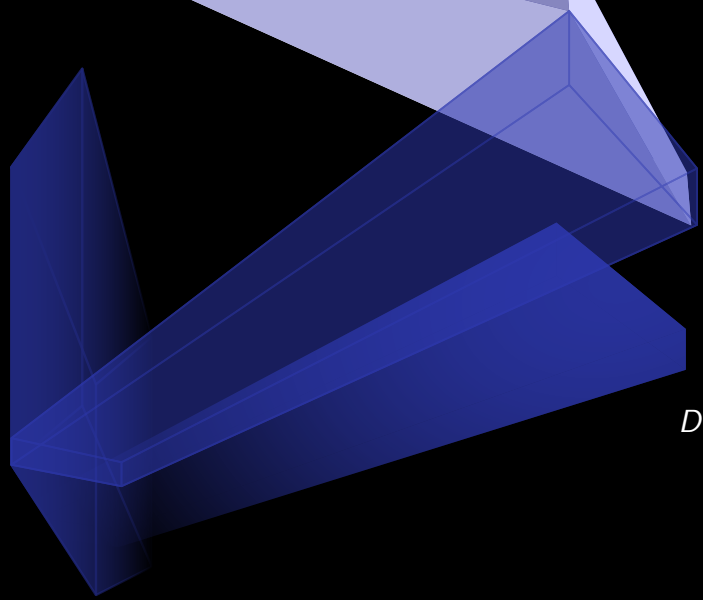


grain

D Language for Deep Learning

ML Meetup KANSAI #3 LT

4. Oct. 2018

D Language for Deep Learning

language

- **like C++**: fast, strongly typed, LLVM/GCC backend
- **like Python**: simple, lightweight, jupyter support

libraries[1]

- **mir**: N-dim fast algorithm, numpy-like APIs
- **dcompute**: CUDA kernel DSL

[1] https://github.com/libmir

grain

deep learning framework for D

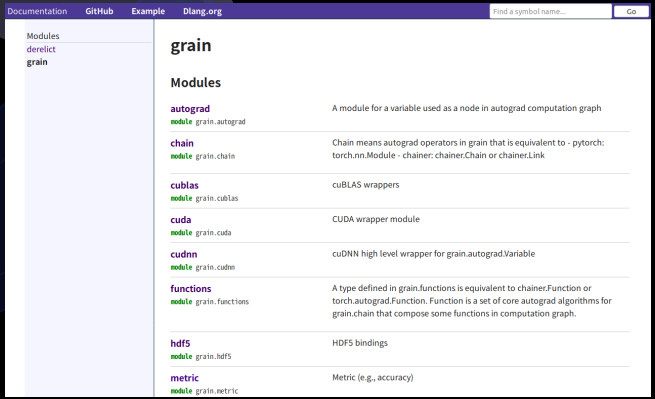
- https://github.com/ShigekiKarita/grain
- boost software license 1.0

philosophy

- **DYNAMIC**: like chainer and pytorch
- **SAFE**: statically typed variable and function
- **LIGHT**: simple like Python, small like C++
- **FAST**: mir and CUDA backend

grain

documentation [2]



[2] https://shigekikarita.github.io/grain/grain.html

grain is dynamic

like chainer ...

```
foreach (epoch; 0 .. 10) {
  foreach (i; niter.permutation) {
    auto xs = inputs[i].variable;
    auto ts = targets[i].variable;
    auto ys = model(xs);
    auto loss = crossEntropy(ys, ts);
    auto acc = accuracy(ys, ts);
    model.zeroGrad();
    loss.backward();
    optimizer.update();
  }
}
```

grain is safe

but statically typed and optimized.

```
foreach (epoch; 0 .. 10) {
  foreach (i; niter.permutation) {
    Variable!(float, 3, HostStorage) xs = inputs[i].variable;
    Variable!(int, 1, HostStorage) ts = targets[i].variable;
    Variable!(float, 2, HostStorage) ys = model(xs);
    Variable!(float, 0, HostStorage)loss =crossEntropy(ys, ts);
    float acc = accuracy(ys, ts);
    model.zeroGrad();
    loss.backward();
    optimizer.update();
  }
}
```

grain is safe

every function is statically typed and optimized.

```
struct Sigmoid(T, size_t dim) {
  Variable!(T, dim, HostStorage) y;

  nothrow forward(Variable!(T, dim, HostStorage) x) {
    auto y = x.sliced.map!(a => tanh(a * 0.5) * 0.5 + 0.5)
              .slice.variable(x.requiresGrad);
    if (x.requiresGrad) this.y = y;
    return y;
  }
  nothrow backward(Variable!(T, dim, HostStorage) gy) {
    auto ys = this.y.sliced;
    return slice((1.0 - ys) * ys * gy.sliced).variable;
  }
  mixin FunctionCommon; // inject type checking
}
```

grain is safe

Chainer/PyTorch issue

```
Python 3.6.5 |Anaconda, Inc.| (default, Apr 29 2018, 16:14:56)
Type 'copyright', 'credits' or 'license' for more information
IPython 6.3.1 -- An enhanced Interactive Python. Type '?' for help.

In [1]: import numpy

In [2]: numpy.zeros((3, 4)) * numpy.zeros((2, 2, 2))
---------------------------------------------------------------------------
ValueError                                Traceback (most recent call last)
<ipython-input-2-d271145d2603> in <module>()
----> 1 numpy.zeros((3, 4)) * numpy.zeros((2, 2, 2))

ValueError: operands could not be broadcast together with shapes (3,4) (2,2,2)
```

grain is safe

Chainer/PyTorch issue

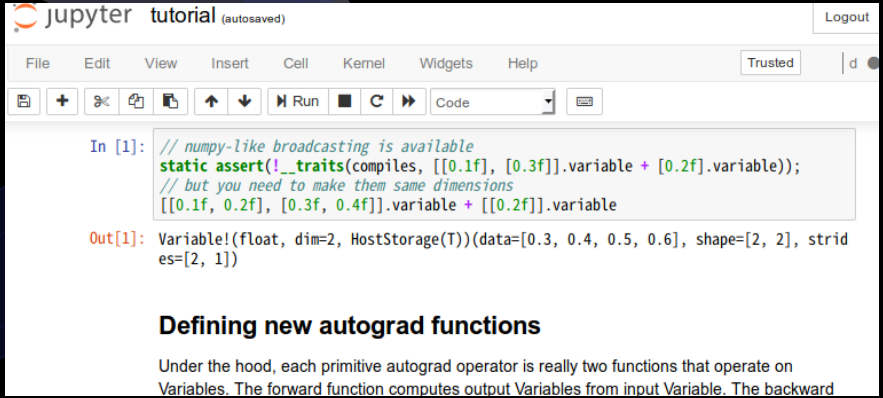
- runtime overhead
  - for-loop, dynamic dispatch, func call
- runtime error:
  - type error, dim mismatch, exception, memory leak

D solution

- template based compile-time code generation (static if/foreach)
- compile-time type/dim/exception checking

grain is a lightweight framework

Jupyter notebook support [3]



[3] https://github.com/ShigekiKarita/grain/blob/master/tutorial.ipynb

grain is a lightweight framework

smaller code and footprint

framework	code lines	lib size [mb]	lib type
grain	**12,431**	**0.6**	**static**
chainer	162,106	6	python code
pytorch	193,754	911	dynamic
tensorflow	130,475	285	dynamic

smaller exe file (MNIST : 1.8MB, CIFAR: 2.3MB)

grain is as fast as other frameworks

task	backend	framework	train iter/sec
mnist	CUDA	grain	270
		chainer	340
		pytorch	200
	CPU	grain	160
		chainer	95
		pytorch	110

- chainer 4.5.0, pytorch 0.4.1, MKL2018, CUDA9, CUDNN7
- pytorch is built from source. modified official scripts to be fair.

grain is as fast as other frameworks

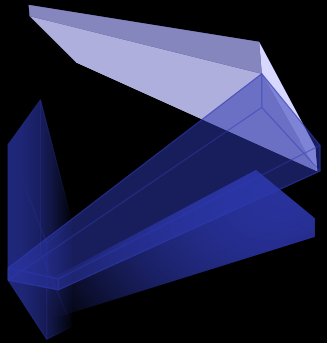
task	backend	framework	train iter/sec
ptb	CUDA	grain	3.1
		chainer	3.4
		pytorch	12
	CPU	grain	1.2
		chainer	2.1
		pytorch	2.4

- chainer 4.5.0, pytorch 0.4.1, MKL2018, CUDA9, CUDNN7
- pytorch is built from source. modified official scripts to be fair.

grain: summary

deep learning framework for D language

- **DYNAMIC**: like chainer and pytorch
- **SAFE**: statically typed variable and function
- **LIGHT**: simple like Python, small like C++
- **FAST**: mir and CUDA backend



Thanks for your attention

https://github.com/ShigekiKarita/grain

examples

- Image recognition (mnist, cifar)
- Language modeling (shakespere, ptb)
- WIP
  - Reinforcement learning (cartpole)
  - Speech recognition (librispeech)
  - Machine translation (anki)

future work

- probabilistic programming
- lazy evaluation mode
- low resource environment (RasberryPi)